ISO RM-ODP Standards: A Reference Model of Open Distributed Processing

Steve Louis

Project Lead, High Performance Storage
National Energy Research Supercomputer Center

Lawrence Livermore National Laboratory
P.O. Box 5509 Mail Code L-561
Livermore, California 94551
louis@nersc.gov

What is RM-ODP?

- A coordinating framework for standardization of open distributing processing
- An architecture created to support
  - distribution
  - interworking
  - interoperability
  - portability
- A big picture to organize pieces of an open distributed system into a coherent whole
- Abstract but not vague
- Components but not implementations

Drivers for RM-ODP

- Application portability across heterogeneous platforms
- Meaningful information exchange throughout the system
- Convenient use of functionality throughout the system
- Distribution transparency for both users and applications programmers
- Ability to describe most open distributed systems available today and in the future

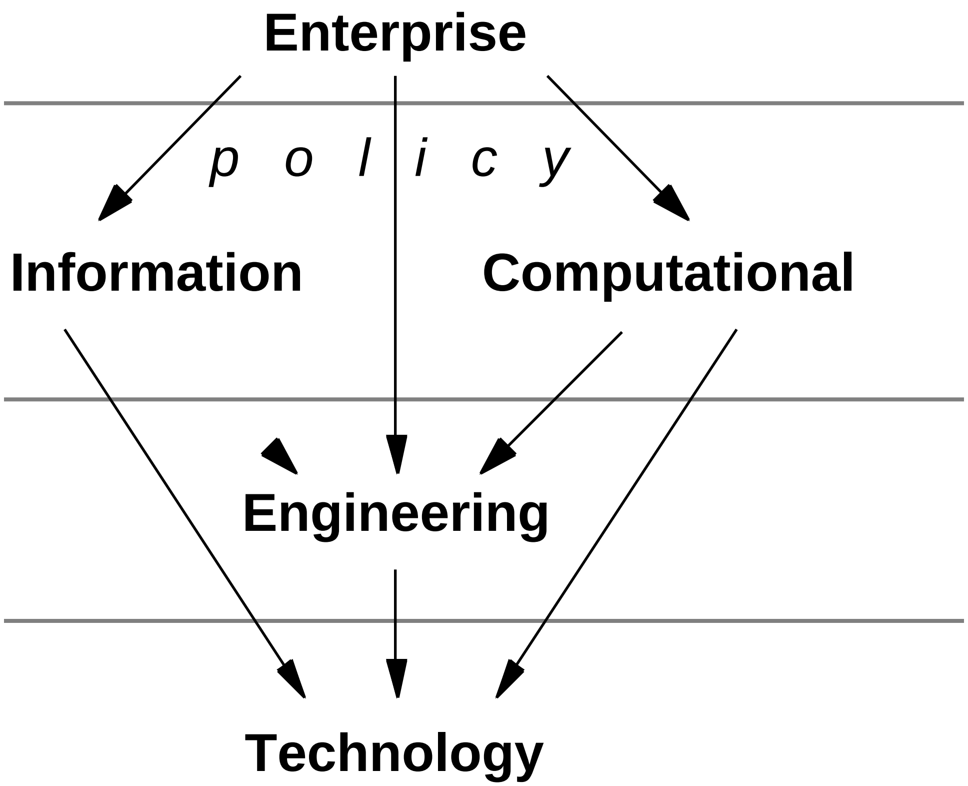
Structure of RM-ODP

- Known as both ISO 10746 and ITU-T X.900
- RM-ODP consists of four parts
  - Part 1: Overview and Guide to Use (ISO 10746-1/ ITU-T X.901) (motivational overview and explanation of key concepts)
  - Part 2: Descriptive Model (ISO 10746-2 / ITU-T X.902) (precise definitions required to specify distributed systems)
  - Part 3: Prescriptive Model (ISO 10746-3 / ITU-T X.903) (framework of ODP concepts, structures, rules, functions)
  - Part 4: Architectural Semantics (ISO 10746-4 / ITU-T X.904) (using formal description techniques to model ODP concepts)
- Parts 2 and 3 are International Standards
- Parts 1 and 4 are Draft International Standards

A Viewpoint Based Framework

- **RM-ODP defines five *viewpoints***
  - ***Enterprise* Viewpoint (purpose, scope, and policies)**
  - ***Information* Viewpoint (information processing semantics)**
  - ***Computational* Viewpoint (functional decomposition)**
  - ***Engineering* Viewpoint (distribution support infrastructure)**
  - ***Technology* Viewpoint (implementation technology choices)**
- **Each viewpoint has its own**
  - **concepts, structures, rules (i.e., a specification language)**
- **Using these viewpoint languages allows large, complex systems to be separated into manageable pieces and coherently specified**

Viewpoints vs. S/W Engineering



Enterprise *Requirements Analysis*

policy

Information **Computational** *Functional Specification*

Engineering *Design*

Technology *Implementation*

Enterprise Viewpoint

- **Used to specify *organizational* requirements**
- **Helps minimize technology-imposed restrictions**
- **Policies can be defined in terms of**
  - ***objects:* both active (managers, data producers/consumers) and passive (e.g., information granules)**
  - ***communities:* object groupings intended to achieve some purpose (e.g., data storage system, DBMS, user groups)**
  - ***roles:* expressed in terms of policies**
    - **permissions (what can be done)**
    - **prohibitions (what must not be done)**
    - **obligations (what must be done)**

Information Viewpoint

- Used to specify *information* required by an application (through the use of schemas)
- Schemas can be
  - *static:* captures object state/structure at particular time
  - *invariant:* restricts object state/structure at all times
  - *dynamic:* defines permitted changes in state/structure
- Schemas can also describe relationships or associations between objects
- A schema can be composed from other schemas to describe complex objects
- Can use conceptual schemas, E-R models, etc.

Computational Viewpoint

- Used to specify *functionality* of an application
- Viewpoint is object-based
  - objects encapsulate data and behaviors
  - objects offer (one or more) interfaces for interaction
- Viewpoint is *distribution transparent* and defines
  - the objects within the system
  - the activities within the objects
  - the interactions between objects
- Objects in a computational specification can be application or ODP infrastructure objects

Engineering Viewpoint

- Used to specify the *design* of the distribution-oriented aspects of a system
- Not concerned with semantics of the application
- Fundamental objects are
  - objects (both computational and infrastructural)
  - channels (corresponds to a binding or binding object)
- Defines basic structural units and rules
  - *cluster:* set of related objects that are always co-located
  - *capsule:* set of clusters, managers, and channel connections
  - *nucleus:* system extensions supporting ODP concepts
  - *node:* computer system

Technology Viewpoint

- **Used to specify an *implementation* of a system and the information required for testing**
- **RM-ODP has very few rules applicable to technology specifications**

ODP Functions

- **Functions expected to be required in ODP systems to support needs of computational and engineering specification languages**
- ***Management functions***
  - **manages engineering structures (e.g., creation/deletion)**
- ***Coordination functions***
  - **provides consistent effects (e.g., events, transactions)**
- ***Repository functions***
  - **maintains data and metadata (e.g., type repositories)**
- ***Security functions***
  - **access control, authentication, audit, etc.**

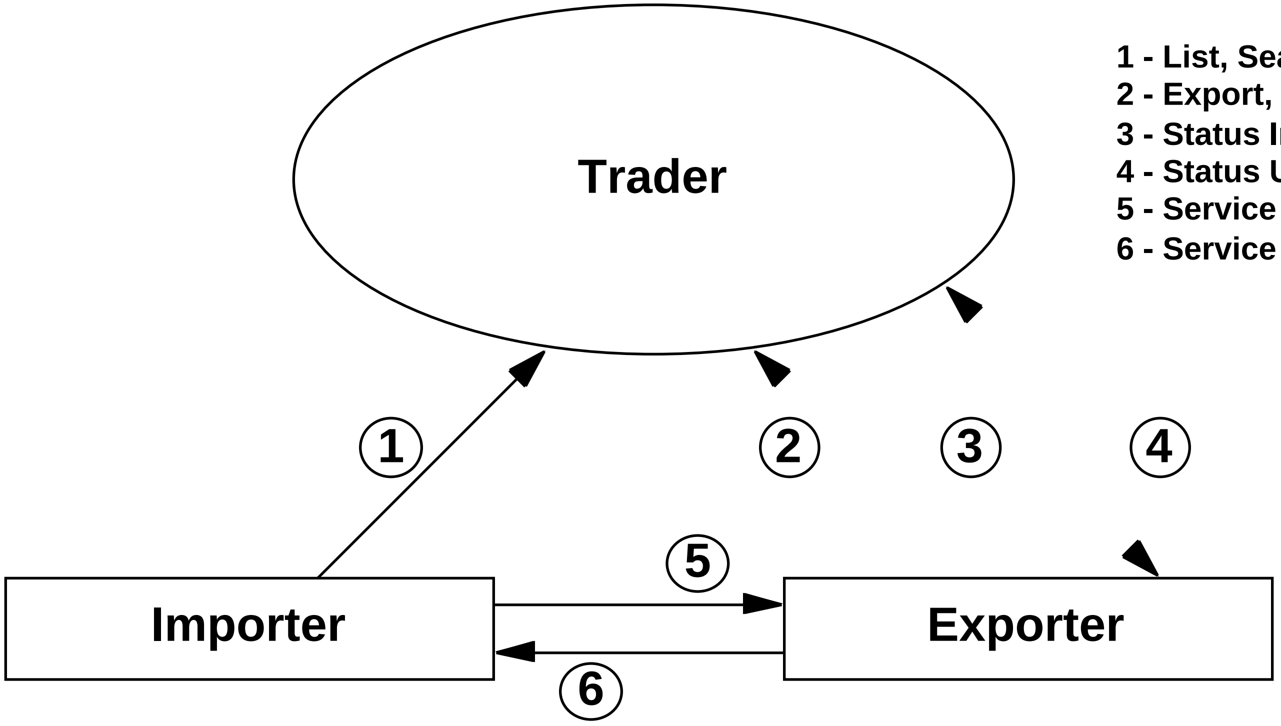
ODP Transparencies

- **Shifts the complexities of distributed systems from applications to the support infrastructure**
  - ***access:*** **hides data representation/procedure differences**
  - ***location:*** **masks use of physical addressing, local/remote**
  - ***relocation:*** **hides object relocation from bound entities**
  - ***migration:*** **masks object relocation from itself**
  - ***persistence:*** **masks deactivation and reactivation**
  - ***failure:*** **masks failures and recoveries for fault tolerance**
  - ***transaction:*** **hides transactional coordination of operations**
  - ***replication:*** **maintains consistency of replicated objects**
- **Above is not necessarily a complete set**

RM-ODP Traders

- **The Trader provides a *dating service* for objects as a repository of service ads**
- **Supports dynamic binding through run-time service discovery**
- **Service providers use the Trader *export* operation to place and modify service ads**
- **Clients use the Trader *import* operation to choose services by specifying required service types and attributes**
- **Subject of current ISO standardization work**

Model for RM-ODP Trading



1 - List, Search, Select
2 - Export, Withdraw, Modify
3 - Status Inquiry
4 - Status Update
5 - Service Invocation
6 - Service Results

Where to Obtain Documents

ftp://ftp.dstc.edu.au/pub/arch/RM-ODP
(Australia, taylor@dstc.edu.au)

ftp://ftp.gmd.de/documents/iso/RM-ODP
(Germany, schoo@fokus.berlin.gmd.d400.de)

ftp://ftp.gte.com/pub/odp
(USA, nicol@gte.com)

Acknowledgment

Much of the information presented here was obtained from a tutorial entitled “Reference Model of Open Distributed Processing (RM-ODP): Introduction” given by Kerry Raymond, CRC for Distributed Systems Technology, during the Third IFIP TC 6 / WG 6.1 International Conference on Open Distributed Processing, Brisbane, Australia, February 1994. Proceedings of this conference were published by Chapman & Hall, ISBN 0-412-71150-8.

Adding Policy to the Draft Archiving Reference Model

Steve Louis

Project Lead, High Performance Storage
National Energy Research Supercomputer Center

Lawrence Livermore National Laboratory
P.O. Box 5509 Mail Code L-561
Livermore, California 94551
louis@nersc.gov

Policies for Digital Archives

- Operations and policy initially suggested by Section 3.2.6 of the draft reference model:
  - costing policies
  - media monitoring for degradation
  - provision for backups
  - product identification
  - interactions with other archives (e.g., federations)
  - preserving information under impending archive dissolution
- Why are policies important for digital archives?
  - loss of data will likely not be tolerated by customers
  - loss of access will likely not be tolerated by customers
  - loss of efficiency will likely not be tolerated by customers

Why Management by Policy?

- Informal ad-hoc management of complex distributed systems will not work
- Human administrators will need help through autonomous execution of management tasks
- Automated and consistent management requires formal specification of policies
- Policies provide information for:
  - *pre-conditions:* "What is necessary to ... ?"
  - *post-conditions:* "What happens if ... ?"
- Generalized policy handling functions are legitimate standardization concerns

What is a Policy?

- **From Hewlett-Packard’s DISA storage model:**
  - **a collection of object classes and managers is a *policy***
- **From RM-ODP Enterprise Viewpoint:**
  - **a set of rules related to a particular purpose is a *policy***
  - **policies can be *permissive, prohibitive or obligatory***
- **From Moffett, Sloman:**
  - **a rule that describes a management activity is a *policy***
  - **policies provide either *authorization or motivation***
  - **motivations can be *positive, negative or both***
- **From Meyer, Popien formal PDN notation:**

<policy>::=“**policy**”<name>“**for**”<domain>“**with**”<behavior_desc>“**end policy**”.
<behavior_change>::=<event_triggered_behavior>**|**<conditional_behavior>.
<event_triggered_behavior>::=“**on**”<external_event>“**=>**“<behavior>.
<conditional_behavior>::=“**if**”<internal_state>“**then**”<behavior>.
<policy_behavior>::=<modality><behavior>.
<modality>::=“**force**”**|**”**permit**”**|**”**prohibit**”**|**...**|**<empty>.

Policies as Object Instantiations

- **Policies may be considered as instantiations of specialized object classes**
- **Members of a policy object class would have the following kinds of attributes:**
  - ***Modality:*** **describes authorization or motivation**
  - ***Policy subjects:*** **defines objects that policy applies to**
  - ***Policy target object:*** **states class policy is directed to**
  - ***Policy goal:*** **defines abstract or specific actions of policy**
  - ***Policy constraints:*** **defines conditions that must be satisfied before the policy can become active**

Generalized Policy Handling

- A general policy handling function should include the following:
  - naming a policy
  - storing a policy
  - modifying a policy
  - replacing a policy
  - merging multiple policies
  - resolving conflicting policies
  - applying refinements to a policy
- Object inheritance can be used for localized policy, and can ensure consistency between inherited local policies and global policies

Management in RM-ODP

- RM-ODP defines four functional areas:
  - Quality of Service (e.g., fault/performance management)
  - Accounting
  - Configuration
  - Policy Definition and Policy Monitoring
- Policies are defined to extend, reduce, or modify the behavior of objects in a domain
- Security and communication are operational areas, not necessarily management concerns

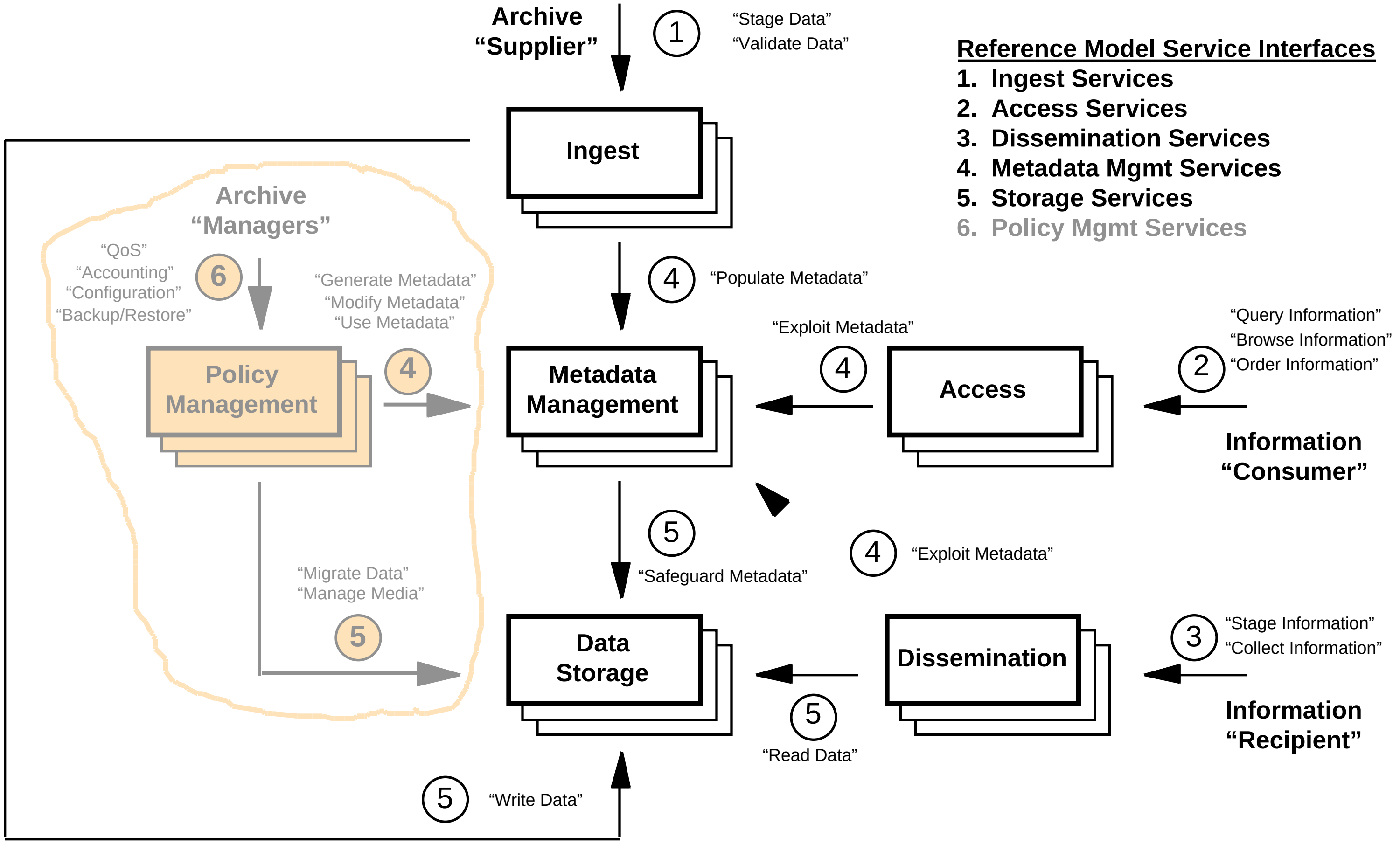
Management Dimensions

- **Policies for management may be categorized by a three-dimensional space model:**
  - **function**
    - **QoS (fault/performance), accounting, configuration**
  - **time**
    - **planning, development, maintenance**
  - **scope**
    - **network, system, application, enterprise**

Policies and Metadata

- Policy descriptions should probably be represented by metadata
- Policy metadata can be generated by:
  - “protein” administrators
  - management applications
- Policy metadata can be modified by:
  - manual processes
  - analysis objects
- Policy metadata can be used by:
  - policy managers (in management applications)
  - policy agents (in components and objects)

Policy in Digital Archive Model



Reference Model Service Interfaces

1. Ingest Services
2. Access Services
3. Dissemination Services
4. Metadata Mgmt Services
5. Storage Services
6. Policy Mgmt Services

Policy References

[1] T. Koch and B. Kramer. Towards a Comprehensive Distributed Systems Management, in K. Raymond and L. Armstrong, eds., *Open Distributed Processing: Experiences with Distributed Environments*, pp. 259-270, Chapman & Hall, 1995

[2] B. Meyer and C. Popien. Flexible Management of ANSAware Applications, in K. Raymond and L. Armstrong, eds., *Open Distributed Processing: Experiences with Distributed Environments*, pp. 271-282, Chapman & Hall, 1995.

[3] M. Sloman and K. Twidle. Domains: A Framework for Structuring Management Policy, in M. Sloman, ed., *Network and Distributed Systems Management*, pp. 433-453, Addison-Wesley, 1994.

[4] J. Moffett. Specification of Management Policies and Discretionary Access Control, in M. Sloman, ed., *Network and Distributed Systems Management*, pp. 455-480, Addison-Wesley, 1994.

[5] R. Baird, S. Karamooz and H. Vazire. *Distributed Information Storage Architecture*, Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems, Monterey, CA, April 1993.

RM-ODP References

[1] ISO/IEC DIS 10746-1, Basic Reference Model of Open Distributed Processing - Part 1: Overview and Guide to Use, 1995.

[2] ISO/IEC IS 10746-1, Basic Reference Model of Open Distributed Processing - Part 2: Descriptive Model, 1995.

[3] ISO/IEC IS 10746-1, Basic Reference Model of Open Distributed Processing - Part 3: Prescriptive Model, 1995.

[4] ISO/IEC DIS 10746-1, Basic Reference Model of Open Distributed Processing - Part 4: Architectural Semantics, 1995.

[5] ISO/IEC JTC1/SC2/WG7, Draft ODP Trading Function, ITU-TS Rec. X.9tr | ISO/IEC 13235, 1994.